World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

A behavioral risk-aware access control framework for secure digital legacy systems

Hampo JohnPaul A.C.[1, *], Kinyuy Marie-Noel Ngala [2], Fomukom Mark Nsah Tanyi [2] and Kpudzeka Blessed Nkurumah [2]

[1] *Software Engineering Department, Faculty of Science and Technology, Catholic University of Cameroon, Bamenda, North West Region, Cameroon.*
[2] *Department of Computer Science, Faculty of Science and Technology, Catholic University of Cameroon, Bamenda, North West Region, Cameroon.*

World Journal of Advanced Research and Reviews, 2026, 31(01), 868–882

Publication history: Received on 31 May 2026; revised on 14 July 2026; accepted on 16 July 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.31.1.1903

Abstract

Digital legacy systems must decide, without the account owner present to confirm intent, whether a request for emergency access to sensitive testamentary documents is legitimate. Existing approaches address this problem only partially: platform-specific legacy-contact features are not interoperable and require no independent corroboration, attribute- and role-based access control models regulate access during active use but are not designed for posthumous or inactivity-triggered release, and machine-learning-based anomaly detection improves suspicious-activity detection at the cost of data and computational requirements that are impractical for a lightweight, single-tenant application. This paper proposes and evaluates a behavioral risk-aware access control framework that unifies four mechanisms within a single architecture: (i) a rule-based behavioral risk-scoring engine that continuously derives a 0-100 risk value from failed-login and high-risk-action signals; (ii) a three-stage, checkpoint-based inactivity/liveness-detection algorithm that escalates from confirmation e-mails to an automatic emergency-access trigger; (iii) a threshold (2-of-3) multi-party authorization protocol that releases access only once independent trusted contacts submit single-use cryptographically random codes; and (iv) role-based access control combined with AES-256-CBC document encryption and full audit logging. We instantiate the framework in a working prototype, LegacyVault, built on Node.js, Express.js, and MySQL, and evaluate it through functional, performance, and security testing aligned to an explicit threat model covering credential compromise, trusted-contact collusion, and insider access. Testing shows that all implemented controls (authentication, encryption, threshold approval, inactivity escalation, and audit logging) operate as designed, with core operations completing in 1.2-7.0 seconds on a local test deployment. We discuss how the framework's design compares with role/attribute-based access control, single-party legacy-contact tools, and blockchain and ML-based alternatives, and we report the scale limitations of the present evaluation transparently rather than overstating a small, single-environment test as a production-grade security guarantee.

Keywords: Behavioral Risk Scoring; Access Control; Inactivity Detection; Multi-Party Authorization; Digital Legacy Management; Threshold Cryptography; Audit Logging

1. Introduction

Digital legacy management concerns how sensitive digital assets, in particular testamentary documents such as wills, are stored, protected, and released to beneficiaries after an owner's death or prolonged incapacity. Traditional physical will storage is vulnerable to loss, damage, tampering, and unauthorized access [1]. Digitizing this process introduces a harder problem than ordinary document security, notably an access-control decision that would normally require the owner's active confirmation must instead be made in the owner's involuntary absence, using only indirect evidence such as inactivity, behavioral signals, and the judgment of trusted third parties; to affirm that the request is legitimate.

* Corresponding author: Hampo JohnPaul A.C

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0.

Conventional access-control models are not designed for this problem. Role-Based Access Control (RBAC) [2,3] and Attribute-Based Access Control (ABAC) [4] regulate who may act on a resource while its owner is an active participant in the system; neither model specifies how access should transition when the subject who would normally authorize a request is permanently or indefinitely unavailable. Consumer platforms have introduced narrow, platform-specific answers to this gap, for example inactivity-based account management and legacy-contact designation; but these tools are not interoperable, and none of the major implementations we reviewed require independent corroboration from more than one trusted party before releasing access [5,6]. Academic proposals go further: blockchain-based inheritance systems automate asset release through smart contracts [7], and AI-based anomaly detection can flag suspicious account activity from learned behavioral baselines [8,9]. Both approaches, however, introduce costs like data and compute requirements in the machine-learning case and infrastructure complexity in the blockchain case. These costs are difficult to justify for a lightweight, individually deployed will-management application.

This gap motivates a different design question: can a lightweight, rule-based framework combine behavioral risk assessment, inactivity-triggered escalation, and distributed human authorization into a single access-control model that is auditable, deployable without specialized infrastructure, and resistant to both credential compromise and single-party collusion? This paper answers that question by proposing a Behavioral Risk-Aware Access Control (BRA-AC) framework and evaluating it through a working prototype, LegacyVault, that was designed and implemented as part of this study.

Framing the problem this way also clarifies why a purely cryptographic or purely procedural solution is, on its own, insufficient. A cryptographic threshold scheme can guarantee that a secret is unrecoverable without a quorum of shares [11], but it says nothing about when that quorum should be permitted to attempt reconstruction in the first place, knowing that timing decision is exactly what inactivity detection is for. Conversely, a purely procedural rule (“notify a next of kin after ninety days of inactivity”) says nothing about whether the request being processed is itself suspicious, which is what a behavioral risk signal is for. The framework proposed here treats these as complementary layers of a single access-control decision rather than as competing solutions, and Sections 4-8 describe how each layer is designed to compose with the others without requiring the others to be replaced.

The contributions of this paper are as follows:

- We define a behavioral risk-aware access control framework that couples a continuous, rule-based risk score with role-based access control and a threshold multi-party authorization protocol, explicitly targeting the posthumous/inactivity access problem that conventional RBAC and ABAC models do not address.
- We formalize the framework's core mechanisms which are a bounded, additive risk-scoring function; a three-checkpoint liveness/inactivity-detection algorithm; and a 2-of-3 threshold authorization protocol using single-use cryptographically random codes; and present them as reusable algorithmic components independent of the specific prototype.
- We instantiate the framework in a working system and evaluate its functional correctness, operational performance, and resistance to a defined set of threats (credential compromise, brute-force login, trusted-contact collusion, unauthorized document access, and audit-trail tampering) through structured testing.
- We report the results, and their limitations, transparently: sample sizes in the present evaluation are small and the test environment is a single local deployment, so the reported figures should be read as evidence of correct mechanism behavior rather than statistically powered security or reliability guarantees.

The remainder of this paper is organized as follows. Section 2 reviews related work on access control models, behavioral risk analytics, inactivity/dead-man's-switch mechanisms, and multi-party authorization. Section 3 defines the threat model and design requirements. Section 4 presents the framework architecture. Sections 5-7 formalize the behavioral risk-scoring engine, the inactivity-detection algorithm, and the multi-party authorization protocol, respectively. Section 8 describes the access-control model and role permissions. Section 9 details the prototype implementation. Section 10 describes the security-evaluation methodology, Section 11 reports results, Section 12 discusses the findings and limitations, and Section 13 concludes the paper.

2. Related Work

2.1. Access control models

Role-Based Access Control assigns permissions to roles rather than individuals, simplifying administration in systems with well-defined organizational structures [2]. The NIST RBAC standard formalizes core, hierarchical, and constrained variants of the model [3]. Attribute-Based Access Control generalizes this by granting access based on attributes of the subject, object, and environment, evaluated against policy rules [4]; NIST's guidance on ABAC notes that this flexibility comes at the cost of policy-authoring complexity. Both models, however, are designed around an implicit assumption: that the resource owner, or an administrator acting in the owner's stead, remains a participant in the system who can be consulted or who has pre-authorized specific roles or attributes. Neither model natively expresses a rule such as "grant this permission only if the owner has been inactive beyond a threshold and a quorum of independent third parties agree," which is precisely the decision a digital legacy system must make.

2.2. Behavioral analytics and risk scoring

User and Entity Behavior Analytics (UEBA) approaches build statistical or machine-learned baselines of normal account behavior and flag deviations as potentially malicious [8]. [8] and [9] show that such systems can achieve strong detection accuracy in cloud environments, but both note the dependence on large training datasets and non-trivial computational resources which is a cost that is disproportionate for a single-tenant application such as a personal will manager, where the volume of behavioral data generated by one account is far too small to train a reliable model. This motivates rule-based, additive risk-scoring approaches as a lighter-weight alternative for low-data settings: rather than learning a baseline, a small number of interpretable signals (for example, repeated failed logins, or a burst of high-risk actions) are combined into a bounded score using fixed weights. This trade adaptability to novel attack patterns for transparency, auditability, and independence from training data are properties that matter for a system whose access decisions must later be explainable to executors, courts, or beneficiaries.

A further consideration specific to digital-legacy systems is that a behavioral score in this setting is not only a security signal but potentially part of the evidentiary basis for a decision with legal and emotional consequences hence, releasing a deceased person's testamentary documents to the wrong party, or failing to release them to the right one, is a materially different failure mode from, say, temporarily locking out a cloud-storage user pending re-authentication. This raises the bar for explainability beyond what is typically required of a UEBA deployment protecting an active enterprise account, and is a further argument, alongside the low-data-volume argument above, for a transparent additive rule rather than an opaque learned model at the individual-account scale considered in this paper.

2.3. Inactivity detection and dead-man's-switch mechanisms

The dead-man's-switch pattern which is an action that triggers automatically unless periodically reset, is a long-standing mechanism for releasing information when a party becomes unavailable, and is the conceptual basis of consumer inactivity-based account tools such as inactive-account managers and legacy-contact designation [5,6]. Existing platform implementations of this idea are binary and single-stage: an account is deemed inactive after one threshold and a single pre-designated contact is notified. [10] observe that the resulting lack of a graduated, multi-checkpoint warning process contributes to inconsistent outcomes across platforms, since a single missed check (which could reflect a temporary loss of e-mail access rather than incapacity) is treated the same as genuine long-term inactivity. A staged approach, in which the system issues successive confirmation requests at defined checkpoints before escalating, reduces the risk of a false trigger from a single missed communication, at the cost of a longer time-to-release in a genuine emergency. This tension notably, between minimizing false triggers and minimizing the delay before a legitimate beneficiary gains access; is analogous to the sensitivity/specificity trade-off familiar from intrusion-detection research, but with the added complication that, unlike a false intrusion alert, a false inheritance trigger cannot simply be dismissed and re-evaluated after the fact without risk of premature document disclosure.

2.4. Multi-party and threshold authorization

Requiring agreement from more than one party before releasing a sensitive resource is a well-established principle in security engineering, most formally expressed in threshold cryptographic schemes such as Shamir's secret sharing, in which a secret is reconstructible only from a quorum of shares [11]. Applied at the workflow level rather than the cryptographic-share level, the same principle underlies dual-control and "four-eyes" procedures in high-assurance environments. In the digital-inheritance literature, [12] find that existing cloud document-storage systems generally lack any multi-party authorization model for conditional emergency access, relying instead on a single administrator or a single designated contact. [7] proposes automating inheritance release through blockchain smart contracts, which can

encode threshold-style release conditions on-chain, but the same study reports that the infrastructure and key-management overhead of a blockchain deployment limits practical adoption for individual users. A workflow-level, application-managed threshold scheme requiring, for example, two of three independently notified trusted contacts to submit distinct verification codes; can achieve a similar resistance to single-party compromise or coercion without requiring distributed-ledger infrastructure, at the cost of relying on the honesty of the hosting application rather than a cryptographically verifiable ledger. This distinction between a cryptographic and a workflow-level threshold guarantee is not merely academic: it determines which adversaries the scheme actually defends against. A workflow-level scheme is well suited to defending against a single compromised or coerced end-user account (a trusted contact whose e-mail or device has been taken over), but it does not, by itself, defend against an adversary who can act at the level of the hosting application or its database, hence a distinction this paper returns to when interpreting the security evaluation in Section 12.2.

2.5. Digital legacy platforms and audit logging

[13] document the fragmented, provider-specific nature of existing digital-death handling on major platforms, and [14] analysis of the legal status of digital assets after death shows that the absence of standardized legal frameworks compounds the technical fragmentation identified above. [15] show that most cloud-storage platforms are designed for security during active use and do not account for inactivity- or death-triggered scenarios. [16] find that audit logging materially improves transparency and incident detection in distributed systems, but that many systems lack tools to analyze log data beyond simple retrieval, and [17] reports that users' trust in a digital system handling sensitive legal information is strongly associated with the presence of clear, auditable activity logs. This is a finding that directly motivates comprehensive audit logging as a core, not optional, component of the framework proposed here.

2.6. Summary of the gap addressed by this work

Table 1 Gap analysis across access-control, behavioral-analytics, and digital-legacy approaches, and the position of the proposed framework.

Approach	Behavioral risk signal	Inactivity handling	Multi-party authorization	Key limitation
RBAC / ABAC [2,3,4]	No	No	No	Fine-grained but assumes an active owner/administrator
UEBA / ML anomaly detection [8,9]	Yes (learned, data-intensive)	No	No	Needs large training data; impractical per-account
Platform legacy-contact tools [5,6]	No	Single-stage, binary	No (single contact)	Not interoperable; no independent corroboration
Blockchain smart-contract inheritance [7]	No	Programmable, but not staged	Possible on-chain, but infra-heavy	High infrastructure/adoption cost
Threshold cryptography (e.g., secret sharing) [11]	No	No	Yes, at the cryptographic layer	Not integrated with behavioral/inactivity signals
Proposed BRA-AC framework (this work)	Yes (rule-based, auditable)	Yes (3-checkpoint, graduated)	Yes (2-of-3, workflow-level)	Lightweight; relies on application-level trust, not a distributed ledger

Table 1 summarizes how existing access-control models, behavioral-analytics approaches, and digital-legacy tools each address only part of the problem that a digital-legacy access-control decision requires. No reviewed approach combines (a) a lightweight, auditable behavioral risk signal, (b) graduated inactivity escalation, and (c) threshold, workflow-level multi-party authorization within a single access-control model. The framework proposed in Sections 4-8 is designed to close this combined gap.

3. Threat Model and Design Requirements

3.1. Assets and trust boundaries

The framework protects two classes of asset: the confidentiality and integrity of stored testamentary documents, and the correctness of the access-control decision that ultimately releases those documents to a beneficiary. Three principal parties interact with the system: the document owner, one or more trusted contacts, and a system administrator. The hosting application (server and database) is trusted to execute the framework's logic correctly; it is not assumed to be resistant to a fully compromised administrator with direct database access, which is a limitation discussed in Section 12.2.

This trust boundary is drawn deliberately, rather than as an oversight, to keep the framework's scope tractable, thus a system that also had to defend against a fully malicious operator of its own infrastructure would need mechanisms for example, client-side encryption with keys never held by the server, or a cryptographic threshold scheme evaluated outside the server's control; that are architecturally quite different from, and more costly to deploy than, the server-side framework evaluated in this paper. We treat the operator/administrator of the hosting application as trusted not to actively subvert the system, while still designing individual controls (Section 9.2) so that routine administrative access does not incidentally expose document contents; this is a narrower but still meaningful notion of insider protection than full protection against a malicious operator, and the difference is made explicit here so that it is not later overstated.

3.2. Adversary classes

- External credential attacker: attempts to guess or brute-force a document owner's password to gain unauthorized access to their account and documents.
- Single malicious or coerced trusted contact: attempts to trigger or approve an emergency-access request alone, without the cooperation or knowledge of other trusted contacts.
- Network eavesdropper: attempts to intercept credentials, verification codes, or documents in transit between client and server.
- Insider/administrative adversary: an administrator or someone with direct access to the underlying database attempts to read document contents or alter audit records.
- Opportunistic/automated attacker: attempts generic web-application attacks such as SQL injection or credential stuffing against exposed endpoints.

3.3. Design requirements

- R1 – Confidentiality at rest: document contents must remain unreadable to anyone without the application-held decryption key, including database administrators.
- R2 – Resistance to credential compromise: authentication must resist brute-force and credential-stuffing attempts through hashing, rate limiting, and expiring tokens.
- R3 – No single point of authorization for emergency access: no individual trusted contact, acting alone, should be able to release documents.
- R4 – Graduated, reversible inactivity handling: the system must distinguish a genuinely inactive/deceased owner from a temporarily unreachable one before irreversibly triggering emergency access.
- R5 – Complete and tamper-evident-in-intent audit trail: every security-relevant action (login, document access, verification-code issuance, approval, denial) must be logged with enough detail to reconstruct the sequence of events.

Sections 4-8 describe how the proposed framework's four mechanisms, which are architecture, behavioral risk scoring, inactivity detection, and multi-party authorization; are designed to satisfy R1-R5. Section 12.2 discusses which requirements are only partially met by the current prototype.

4. Proposed Framework Architecture

The BRA-AC framework is organized as a layered architecture (Figure 1), instantiated in this study as the LegacyVault prototype. A presentation layer (HTML, CSS, JavaScript, Bootstrap) serves interfaces to three actor types: document owners, trusted contacts, and administrators. An application layer, built on Node.js with Express.js, implements the framework's logical modules: authentication, document management, encryption, behavioral risk scoring, inactivity/liveness detection, multi-party verification, notification, audit logging, and administrative monitoring. A security layer, applied horizontally across the application layer, provides password hashing, symmetric document

encryption, role-based access enforcement, and rate limiting. A database layer (relational, sixteen tables in the prototype) persists users, documents, trusted contacts, access requests, verification codes, approvals, and audit logs, with foreign-key constraints enforced at the database level. An external notification service delivers e-mail alerts for liveness checks, verification codes, and access outcomes.

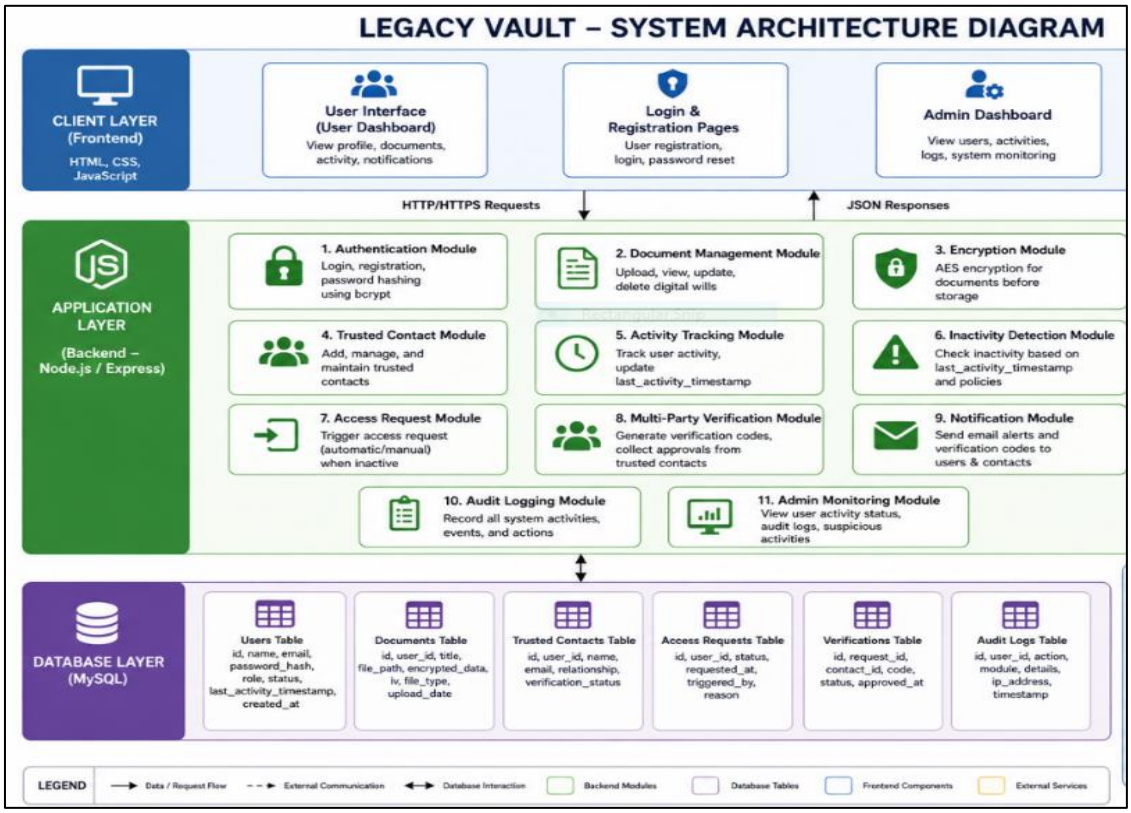

Figure 1 Layered architecture of the Behavioral Risk-Aware Access Control (BRA-AC) framework, as instantiated in the LegacyVault prototype.

Three properties of this architecture are relevant to the requirements in Section 3.3. First, encryption and key management are confined to the application layer: the database layer never receives an unencrypted document, and administrators with direct database access cannot recover document contents without the application secret (R1). Second, the risk-scoring, inactivity-detection, and multi-party-verification modules are functionally independent of one another but share a common audit-logging sink, so that a decision made by any one module is traceable in the same log used to evaluate the others (R5). Third, the architecture separates the workflow-level authorization decision (made by the application layer, based on approvals from trusted contacts) from the cryptographic operation that releases the document (also made by the application layer, but only after the workflow decision succeeds). This separation allows the framework to add or change the authorization protocol described in Section 7 without altering the encryption mechanism described in Section 9.2, and vice versa.

5. Behavioral Risk-Scoring Model

5.1. Design rationale

Section 2.2 argued that machine-learned behavioral baselines are poorly suited to a single-tenant application with limited historical data per account. The framework instead uses a bounded, additive, rule-based risk function that combines two observable signals: recent authentication failures and recent high-risk actions. Both signals are drawn directly from data the system already logs for audit purposes (Section 9.4), so the risk score requires no additional data collection and remains fully explainable, hence each point contributing to a user's score corresponds to a specific, retrievable log entry.

5.2. Formalization

Let f(u) be the number of failed login attempts recorded for user u in the preceding 24 hours, and let h(u) be the number of high- or critical-risk actions recorded for user u in the preceding 7 days. The prototype implementation defines the risk score $R(u) \in [0,100]$ as:

$$R(u) = \min(50, 10 \times \min(f(u), 5)) + \min(50, 15 \times \min(h(u), 3))$$

That is, failed logins contribute up to 50 points (10 points each, saturating after 5 failed attempts), and high-risk actions contribute up to 50 points (15 points each, saturating after approximately 3 events), so that R(u) is bounded in [0,100] regardless of how many additional failures or risk events accumulate beyond the saturation point. The score is recalculated whenever new activity is logged, written back to the user's record, and surfaced on the administrator dashboard so that accounts crossing an operationally meaningful threshold can be reviewed. Because each term saturates rather than growing without bound, a single extended brute-force burst and a sustained one contribute similarly to R(u); this is a deliberate simplification that trades sensitivity to attack duration for a score that remains interpretable and bounded, and is revisited as a limitation in Section 12.2.

5.3. Interpretation and use

R(u) is advisory rather than a hard access-control gate in the current prototype thus it is displayed to administrators to prioritize manual review, and it contributes contextual information alongside but does not by itself trigger or block, hence the inactivity-detection and multi-party-authorization mechanisms described in Sections 6-7. This is consistent with requirement R5 (traceability) but means the framework, as currently implemented, does not yet close the loop between a high risk score and an automated protective action (for example, automatically requiring an additional approval, or temporarily freezing an account, once R(u) exceeds a defined threshold). We identify this as a natural extension in Section 13.

6. Inactivity Detection and Liveness Verification

6.1. Three-stage checkpoint algorithm

The framework detects prolonged inactivity through a scheduled process that runs once every 24 hours and divides a configured inactivity threshold T into three equal checkpoints at T/3, 2T/3, and T. For an illustrative threshold of T = 150 days, the checkpoints fall at day 50, day 100, and day 150. At each checkpoint, if the owner has not logged in or otherwise confirmed activity since the previous checkpoint, the system generates a cryptographically random, 32-byte, hex-encoded confirmation token and e-mails it as a liveness-confirmation link. Clicking the link resets the owner's missed-checkpoint counter to zero; logging in through the normal application interface at any point also resets the inactivity timer. If all three checkpoints are missed, that is, the owner does not respond to any of the three successive confirmation requests over the full threshold period, then the emergency-access workflow described in Section 7 is triggered automatically and trusted contacts are notified.

```
every 24 hours:
  for each active owner u:
    if last_activity(u) is within current checkpoint window:
      continue
    else:
      issue confirmation token(u); email liveness-check link
      missed_checkpoints(u) += 1
      if missed_checkpoints(u) == 3:
        trigger_emergency_access_workflow(u)
  on login(u) or on confirmation link click(u):
    missed_checkpoints(u) = 0; reset last_activity(u)
```

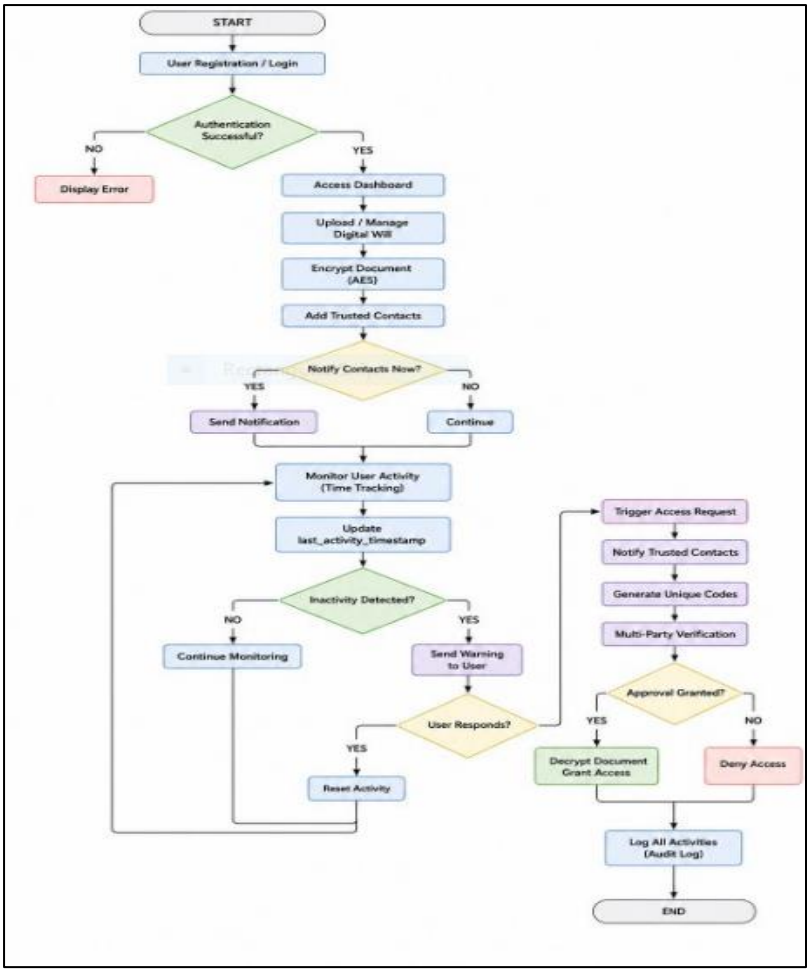

Figure 2 End-to-end inactivity-detection and emergency-access workflow.

6.2. Design trade-offs

The three-checkpoint structure directly addresses design requirement R4 by distinguishing a single missed communication (which could result from a temporary e-mail outage or a missed notification) from sustained inactivity across three independent, spaced confirmation attempts. This graduated structure improves on the single-stage, binary inactivity handling used by the platform tools discussed in Section 2.3 [5,6], at the cost of a longer minimum time-to-release: a genuine emergency cannot be resolved faster than the full threshold period T, even if trusted contacts are ready to act immediately. The appropriate value of T is a policy choice outside the scope of this paper's technical evaluation and would, in a real deployment, likely need to be configurable per owner and reviewed against jurisdiction-specific expectations for how quickly a will should become accessible.

7. Multi-Party Authorization Protocol

7.1. Threshold scheme

Once the inactivity-detection algorithm (Section 6) or a manually initiated request triggers emergency access, the framework applies a 2-of-3 threshold authorization protocol. Up to three trusted contacts are notified [18,19]; each receives a unique 6-digit verification code generated using a cryptographically secure random-number generator over the range 100000-999999. Each code is stored server-side, linked to both the specific access request and the specific contact who received it, and is marked single-use: once submitted, it cannot be reused. As contacts respond, the system atomically counts confirmed approvals against the required threshold of two. The moment a second independent approval is recorded, the request status transitions to approved and all contacts are notified of the outcome; a single denial from any contact immediately closes the request as denied, without waiting for the remaining contact to respond.

```
on emergency_access_triggered(owner):
  contacts = trusted_contacts(owner)        // up to 3
  for each contact in contacts:
    code = secure_random(100000, 999999)
    store(code, request_id, contact_id, used=false)
    email(contact, code)

on contact_submits(code, decision):
  if code.used: reject
  mark code.used = true
  if decision == deny:
    request.status = denied; notify_all(contacts); stop
  if decision == approve:
    request.approvals += 1
    if request.approvals >= 2:
      request.status = approved; notify_all(contacts)
      decrypt_and_release(document)
```

7.2. Security properties

This protocol satisfies design requirement R3: because release requires agreement from at least two of the up-to-three independently notified contacts, a single compromised, malicious, or coerced contact cannot unilaterally release a document, and a single denial is sufficient to block release even if other contacts have not yet responded, which limits the damage of a single account takeover among the contacts. The single-use property of each code prevents replay of an intercepted or previously submitted code. Because approval counting is performed atomically at the point a code is submitted, the protocol also avoids a race condition in which two contacts submitting approvals concurrently could otherwise leave the request in an inconsistent state.

The protocol's security nonetheless depends on the application server correctly enforcing the threshold and single-use logic described above; unlike a cryptographic threshold scheme such as Shamir secret sharing [11], in which the secret is mathematically unreconstructable without a quorum of shares, the guarantee here is a workflow-level guarantee enforced in application logic. An administrator with direct database access could, in principle, alter the stored approval count or mark a request approved without genuine quorum. This is a limitation shared with most non-cryptographic multi-party workflows and discussed further in Section 12.2.

8. Role-Based Access Control Model

The framework layers role-based access control beneath the risk-scoring, inactivity-detection, and multi-party-authorization mechanisms described above, so that even a successfully authenticated user is restricted to actions permitted by their assigned role. The prototype's documented role set comprises document owners (system users), trusted contacts, and administrators; Table 2 summarizes the permissions associated with each role as implemented in the RBAC module.

Table 2 Role permission matrix implemented in the prototype's RBAC module**.**

Capability	Document owner	Trusted contact	Administrator
Register / authenticate	**Yes**	**Yes (via emergency-access flow only)**	**Yes**
Upload / encrypt own documents	Yes	No	No
View own documents	Yes	No (until access approved)	No (no decryption key)
Be assigned as a trusted contact	N/A	Yes	N/A
Submit verification code / approval	No	Yes	No

View own audit log entries	Yes (own account)	No	N/A
View system-wide audit logs / flagged accounts	No	No	Yes
Decrypt document contents	Yes (own documents)	Only after threshold approval	No (application holds the key, not the admin)

One inconsistency in the source project materials should be noted for correction before any further publication: the system-evaluation and discussion sections of the underlying project describe a three-way role separation among "administrators, standard users, and business users," whereas the RBAC module's own implementation description defines only two enforced roles which are, standard user and administrator; with trusted contacts handled as a distinct, non-authenticated actor type rather than a formal account role. Table 2 reflects the roles that are actually documented as implemented; the "business user" role appears to be either a terminology error or an unimplemented design intention in the original project and should be clarified or removed.

Beyond this correction, it is worth noting what the role model in Table 2 deliberately does not do: it does not grant the administrator role any path to document content, even under emergency conditions, and it does not grant the trusted-contact role any access outside the multi-party-authorization workflow described in Section 7. This is a narrower permission set than many enterprise RBAC deployments would use for an equivalent "support" or "custodian" role, and the narrowness is intentional: because the framework's stated threat model (Section 3.2) includes an insider/administrative adversary, the RBAC layer is designed to make the administrator role structurally incapable of the one action, that is, document decryption but that would matter most if that role were compromised or misused, rather than relying on procedural policy alone to discourage misuse.

9. Prototype Implementation

9.1. Technology stack

The framework was instantiated as a working prototype (LegacyVault) using Node.js (≥18) with Express.js (v4.18) for the application layer, MySQL (accessed via the mysql2 driver) for the database layer across sixteen relational tables with enforced foreign-key constraints, and HTML/CSS/JavaScript with Bootstrap for the presentation layer

9.2. Cryptographic mechanisms

Documents are encrypted with AES-256 in Cipher Block Chaining mode (AES-256-CBC) prior to database storage. The 256-bit encryption key is derived from the application secret using the scrypt key-derivation function [20]. A fresh 16-byte initialization vector (IV) is generated per document using a cryptographically secure random-number generator, so that identical files produce different ciphertext; the IV is stored alongside the encrypted, Base64-encoded document data and used to reverse the encryption on authorized download. Administrators cannot decrypt documents through the administrative interface because decryption requires the application-held AES key rather than any credential available to an administrator account, directly supporting requirement R1. Passwords are hashed with bcrypt at a cost factor of 12 (4,096 internal iterations) with a per-password random salt automatically embedded by the algorithm, and non-reversible integrity checks (for example, of access-token identifiers) use SHA-256 via Node.js's built-in crypto module, chosen over bcrypt for these checks because speed, not resistance to offline guessing, is the relevant property when the hashed value is a random token rather than a user-chosen secret.

9.3. Authentication and session management

Following successful login, the backend issues a JSON Web Token (JWT) containing the user's identifier, e-mail, UUID, and role, signed with the HS256 algorithm. The token is attached to the Authorization header of subsequent requests and verified on every protected route by signature and expiry, without a database lookup on each request. Standard-user tokens expire after 24 hours; administrator tokens expire after 8 hours, a shorter window intended to reduce the exposure window for the more privileged role. Login attempts are rate-limited to 5 per 15 minutes per IP address, general API requests to 100 per 15 minutes, and password-reset requests to 3 per 15 minutes, implemented through Express middleware together with a dedicated login-attempts table used for brute-force detection. The Helmet middleware sets standard protective HTTP headers, and all database queries are parameterized to prevent SQL injection.

9.4. Data model and audit logging

The relational schema separates users, documents, trusted contacts, access requests, verification codes, access approvals, and audit/activity logs into distinct tables linked by enforced foreign keys (for example, documents and trusted_contacts both cascade-delete on their owning user, while a document's assigned-contact reference is set to null, rather than cascading, if that contact is removed). Every security-relevant action, namely logins, document access, verification-code issuance and submission, and administrative actions; is written to an activity-log table, together with the contextual detail and risk level used by the risk-scoring function described in Section 5. This continuous, structured log is what allows the administrator dashboard to surface both individual flagged accounts and a reconstructable history of any single access request, addressing requirement R5.

9.5. Notification and administrative monitoring

A dedicated notification module sends e-mail messages at each point where a human actor needs to respond or be informed: emergency-access verification codes to trusted contacts, access-granted and access-denied outcome notifications, welcome messages when a new trusted contact is registered, the three-stage liveness-check e-mails described in Section 6.1, and password-reset links. Coupling notification dispatch directly to the same backend transaction that writes the corresponding database record (for example, writing a new access request and sending the associated verification codes within the same request-handling flow) reduces the chance that a state change is recorded without the affected parties being informed, or vice versa. The administrator dashboard consumes the same audit-log and risk-score data described in Sections 5 and 9.4 to present flagged accounts, recent activity, and the status of in-progress access requests, giving the administrator role a monitoring rather than a document-access capability, consistent with the permission matrix in Table 2.

10. Security Evaluation Methodology

The prototype was evaluated using functional, performance, and security testing, structured to probe the design requirements and adversary classes defined in Section 3. Functional testing exercised each module (registration, login, document upload and encryption, trusted-contact assignment, verification-token validation, and audit logging) with representative valid and invalid inputs. Security testing specifically targeted: authentication correctness under valid and invalid credentials (probing R2, against the external credential-attacker class); encrypted-document protection and administrator inability to read raw document content (probing R1, against the insider/administrative adversary); and the system's ability to block unauthorized access attempts and correctly record them in the audit log (probing R5, against the opportunistic/automated attacker class). Usability testing observed users performing core tasks such as logging in and uploading documents. All testing was conducted on a single local development deployment; the study did not include a field trial, a multi-site deployment, adversarial penetration testing by an independent third party, or a statistically powered evaluation of the risk-scoring function's false-positive or false-negative rate against real attack traffic, points we return to as limitations in Section 12.2.

11. Results

11.1. Functional test results

Table 3 Functional test cases and results

Test case	Expected result	Actual result	Status
Valid user login	Access granted	Access granted	Passed
Invalid password login	Access denied	Access denied	Passed
Document upload (AES-256-CBC encryption)	Successful, encrypted upload	Successful, encrypted upload	Passed
Trusted-contact assignment	Contact assigned	Contact assigned	Passed
Verification-token validation (2-of-3 threshold)	Access granted on 2nd approval	Access granted	Passed
Unauthorized access attempt	Access blocked	Access blocked	Passed
Audit-log recording	Activity logged	Activity logged	Passed

Table 3 summarizes the outcomes of functional test cases across the framework's core modules.

11.2. Performance results

Average response times for core operations, measured on the local test deployment, are reported in Table 4.

Table 4 Average response time by operation (single local test environment; not a load or scalability benchmark).

Operation	Average response time (s)
User login	1.2
Document upload (unencrypted transfer)	2.1
Encrypted document upload (AES-256-CBC)	3.0
Document retrieval and decryption	1.4
Verification-token validation	7.0

11.3. Authentication and threshold-authorization testing

Authentication testing recorded 6 successful logins out of 6 valid-credential attempts and 0 successful logins out of 5 invalid-credential attempts. Threshold multi-party verification testing recorded correct validation (access granted precisely upon the second independent approval, and not before) in 2 of 2 test cases. As with the manuscript's other quantitative figures, these counts are small; they demonstrate that the mechanisms behave as specified on the cases exercised, but they are not a statistically powered estimate of authentication error rates or threshold-protocol reliability under adversarial load, and should not be cited as such.

11.4. Threat-to-control mapping

Table 5 maps each adversary class defined in Section 3.2 to the framework control intended to address it and the corresponding test evidence from this evaluation.

Table 5 Mapping between the threat model (Section 3.2), the framework's controls, and the evidence produced by this study's testing. Several rows rely on design inspection rather than adversarial testing, as noted.

Adversary class	Primary control(s)	Test evidence in this study
External credential attacker	bcrypt hashing (cost 12) + login rate limiting (5/15 min) + JWT expiry	Invalid-password test passed (0/5 successful); rate-limiting not separately load-tested
Single malicious/coerced trusted contact	2-of-3 threshold authorization; single-use codes	Verification-threshold test passed (2/2); collusion of 2 contacts not testable without a live multi-party adversarial trial
Network eavesdropper	Bearer-token authentication over HTTP/HTTPS as configured	Not explicitly verified in this evaluation — TLS enforcement should be confirmed and documented before deployment
Insider/administrative adversary	AES-256-CBC encryption with application-held key; admin interface has no decrypt capability	Confirmed by design/code inspection; not verified through an independent penetration test
Opportunistic/automated attacker (SQLi, unauthorized access)	Parameterized queries; Helmet security headers; RBAC route guards	Unauthorized-access test passed (blocked and logged); no automated vulnerability scan reported

12. Discussion

12.1. Interpretation

The results in Section 11 indicate that each of the framework's core mechanisms namely authentication and password hashing, AES-256-CBC document encryption, the 2-of-3 threshold authorization protocol, and audit logging; functions correctly on the cases tested, which supports the design's basic soundness against the requirements in Section 3.3. The threshold-authorization result is particularly relevant to the framework's central contribution because access was granted precisely on the second independent approval and not before, the tested prototype demonstrates the specific property that motivated the design therefore, avoiding a single point of authorization for emergency access (R3) rather than only demonstrating that authentication in general works.

Relative to the related work reviewed in Section 2, this positions the BRA-AC framework as a middle ground between the single-stage, single-contact designs of existing platform tools [5,6] and the higher-assurance but higher-overhead alternatives of blockchain-encoded release conditions [7] or learned behavioral baselines [8,9]. The framework's rule-based risk score and staged inactivity detection are deliberately simpler and less adaptive than a machine-learned anomaly detector, trading detection sophistication for auditability and independence from training data. This is a trade-off that is appropriate for a personal, single-tenant application but would need re-evaluation for a multi-tenant deployment with a large, heterogeneous user base, where a learned baseline might become both feasible and beneficial.

It is also worth situating the graduated inactivity-detection design (Section 6) against the binary approach used in existing platform tools [5,6]. The three-checkpoint structure directly targets a specific failure mode of binary designs without requiring any additional infrastructure beyond scheduled jobs and e-mail delivery already present in the system. In a binary design, a single missed notification being treated identically to sustained, months-long inactivity. This suggests that meaningful improvements over current platform practice do not necessarily require the heavier architectural changes associated with blockchain- or ML-based proposals; some of the fragmentation and single-stage-triggering problems identified in Section 2 can be addressed through comparatively simple workflow redesign, provided that redesign is evaluated as rigorously as the more novel mechanisms it is compared against.

12.2. Limitations

- Scale and adversarial realism of the evaluation. All results in Section 11 come from a single local development deployment with small trial counts (6 valid and 5 invalid login attempts; 2 threshold-authorization test cases). No independent penetration test, load test, or adversarial red-team exercise was conducted. The threat-to-control mapping in Table 5 therefore rests partly on design/code inspection rather than empirical adversarial testing, and this distinction should not be elided in any subsequent summary of this work.
- The risk score is advisory, not enforcing. As noted in Section 5.3, the current prototype computes and displays R(u) but does not yet use it to automatically gate or add friction to access decisions; the behavioral signal and the authorization workflow are not yet closed-loop.
- Workflow-level, not cryptographic, threshold guarantee. As discussed in Section 7.2, the 2-of-3 authorization protocol's security depends on correct server-side enforcement rather than a cryptographic threshold scheme; a compromised administrator with direct database access could in principle bypass it, which the current design does not prevent.
- No biological death or legal verification. Emergency access is triggered by inactivity and corroborated by trusted-contact approval, not by any medical, civil-registry, or legal confirmation of death or incapacity; the mechanism is a technical proxy, not a legal determination, and any real deployment would need to make this distinction explicit to users and beneficiaries.
- Unconfirmed transport security. The evaluation in this study did not explicitly verify that transport-layer encryption (HTTPS/TLS) is enforced end-to-end in the deployed prototype; this should be confirmed and, if absent, treated as a required addition rather than an optional hardening step, given that credentials and verification codes are transmitted through the same channel.
- Role-model inconsistency. As noted in Section 8, the source project materials describe a three-way role separation in some sections that is not reflected in the RBAC module's own implementation description; this should be resolved before the role model is presented as final in any future version of this work.
- Dependence on trusted-contact reachability. As with any notification-based scheme, the multi-party authorization protocol's timeliness depends on trusted contacts having working e-mail access and reasonable response times, which the framework cannot control or guarantee.

12.3. Implications for practice and research

For practitioners building similar systems, these results suggest that a workflow-level threshold-authorization scheme, combined with graduated inactivity detection, can meaningfully reduce single-party authorization risk without the infrastructure cost of a blockchain deployment, provided the limitations above are acknowledged rather than obscured. Note that a major limitation is particularly the reliance on correct server-side enforcement and the absence of independent adversarial testing. For researchers, the natural next steps are to (a) close the loop between the behavioral risk score and the authorization workflow, for example by requiring an additional approval or a cooling-off period when R(u) exceeds a defined threshold; (b) evaluate the framework under adversarial and load conditions rather than functional correctness alone; and (c) investigate whether a cryptographic threshold scheme (Section 2.4) could replace or complement the current application-level enforcement to remove the administrator-bypass limitation identified above; (d) investigate the absence of biometric identification and authentication [21] and the methodology used for memorialization and legacy contract [22].

13. Conclusion and Future Work

This paper proposed a Behavioral Risk-Aware Access Control (BRA-AC) framework for digital legacy systems, combining a bounded, rule-based behavioral risk score, a three-stage inactivity/liveness-detection algorithm, a 2-of-3 threshold multi-party authorization protocol, and role-based access control with AES-256-CBC document encryption and comprehensive audit logging. We instantiated the framework in a working prototype and evaluated it against an explicit threat model covering credential compromise, trusted-contact collusion, network interception, insider access, and generic web-application attacks. Functional and security testing showed that the framework's core mechanisms operate as designed on the cases tested, and in particular that the threshold-authorization protocol correctly withholds access until a genuine second, independent approval is recorded, thus directly demonstrating the framework's central design goal of eliminating single-party control over emergency access.

The evaluation reported here is a proof-of-concept on a single local deployment with small trial counts, not a production-scale security audit, and several concrete limitations should be addressed before the framework is presented as deployment-ready. These concrete limitations are an advisory rather than enforcing risk score, a workflow-level rather than cryptographic threshold guarantee, unconfirmed transport-layer security, and the absence of independent adversarial testing. Future work should prioritize: closing the loop between the behavioral risk score and access decisions; independent penetration testing and load testing at realistic scale; exploring cryptographic threshold schemes as a stronger alternative or complement to the current application-enforced protocol; machine-learning-based risk scoring once sufficient behavioral data is available in a multi-tenant deployment; and interoperability with legal or civil-registry systems to move inactivity-based inference toward verified legal confirmation of death or incapacity.

More broadly, this study suggests that the access-control problem posed by digital legacy systems is best treated as a composition problem rather than a search for a single dominant mechanism. Behavioral risk scoring, staged inactivity detection, threshold authorization, and role-based access control each address a different facet of the same underlying question, which is whether a request made in the owner's absence should be trusted; and the framework's contribution lies as much in specifying how these mechanisms compose and share an audit trail as in any one mechanism taken alone. We hope the formalizations in Sections 5-7, together with the honestly reported limitations in Section 12.2, provide a reusable and falsifiable starting point for that composition, rather than a claim that the problem is now solved.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Bellamy C., Arnold, M., Gibbs, M., et al. (2013) Life beyond the timeline: Creating and curating a digital legacy. Proceedings of the Community Informatics Research Network (CIRN) annual conference, Prato, Italy.

[2] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. IEEE Computer, 29(2), 38–47. https://doi.org/10.1109/2.485845

[3] Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security, 4(3), 224–274.

[4] Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., & Scarfone, K. (2014). Guide to Attribute Based Access Control (ABAC) Definition and Considerations (NIST Special Publication 800-162). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-162

[5] Apple Inc. (n.d.). Legacy Contact for Apple ID. Apple Support. https://support.apple.com/en-us/HT212360

[6] Google. (n.d.). Inactive Account Manager. Google Support. https://support.google.com/accounts/answer/3036546

[7] Deng, R. (2025). Research on Decentralized Digital Inheritance Management System Empowered by Blockchain and Smart Contracts: With a Discussion on the Inheritance and Destruction of Digital Assets. Preprints. https://doi.org/10.20944/preprints202506.0292.v1

[8] Goswami, M.J. (2024). AI-Based Anomaly Detection for Real-TimeCybersecurity. International Journal of Research and Review Techniques, Volume 3, Issue 1, 45-53

[9] Anugera, E.S. & Mirza, A. (2026). A Hybrid CNN-LSTM Approach for Detecting Cross-Site Scripting Attacks in Web Applications. bit-Tech, 2026, 8 (3), 3508-3518

[10] Feng, Z., Li, Z., Cui, H., & Whitty, M. T. (2025). Identity Management Systems: A Comprehensive Review. Information, 16(9), 778. https://doi.org/10.3390/info16090778

[11] Shamir, A. (1979). How to share a secret. Communications of the ACM, 22(11), 612–613. https://doi.org/10.1145/359168.359176

[12] Musa, A., & Bello, T. (2021). Cloud-based storage systems for legal documents and inheritance challenges. International Journal of Cloud Computing Research.

[13] Öhman, C., & Watson, D. (2019). Are the dead taking over Facebook? A big data approach to digital death. Big Data & Society, 6(1). https://doi.org/10.1177/2053951719842540

[14] Harbinja, E. (2017). Legal aspects of the transmission of digital assets on death. Doctorate Thesis, University of Strathclyde. Repository: https://stax.strath.ac.uk/concern/theses/k3569438f

[15] Mohammad, Z.R. (2025). Cloud Storage Security: Risks and Solutions. Preprints. https://doi.org/10.20944/preprints202511.1778.v1

[16] Di Pilla, P., Pareschi, R., Salzano, F. & Zappone, F. (2023) Listening to what the system tells us: Innovative auditing for distributed systems. Frontiers of Computer Science. 4:1020946. Doi: 10.3389/fcomp.2022.1020946.

[17] Nieuwenhuizen, E. (2025). "8: Trust and transparency in algorithmic governance: a multi-level framework". In the Handbook on Trust in Public Governance. Cheltenham, UK: Edward Elgar Publishing. Retrieved Jul 11, 2026, from https://doi.org/10.4337/9781802201406.00013.

[18] Stallings, W. (2022). Cryptography and Network Security: Principles and Practice (8th ed.). Pearson.

[19] Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero Trust Architecture (NIST SP 800-207). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

[20] Obiri-Tetteh, J.A., Turkson, R.E., Frimpong, E.A., et al. (2025) Bcrypt-Based Optimized Password Hashing Against Targeted Internet of Things Attacks. Cureus Journal of Comput Science. DOI https://doi.org/10.7759/s44389-025-05167-y

[21] Kakade, S. & Raut, U. (2026). Biometric authentication systems: Trends, challenges, and future prospects – A comprehensive review. MethodsX. Volume 16, https://doi.org/10.1016/j.mex.2026.103908

[22] Meta Platforms Inc. (n.d.). Facebook memorialization and legacy contact. Facebook Help Center. https://www.facebook.com/help